BU-CCS-941002
chem-ph/9411003
To Appear in *Computers in Physics* (1994)

# Dimensional Perturbation Theory on the Connection Machine

Timothy C. Germann and Dudley R. Herschbach
*Department of Chemistry, Harvard University, 12 Oxford Street, Cambridge, MA 02138*

Bruce M. Boghosian
*Center for Computational Science, Boston University, 3 Cummington Street, Boston, MA 02215*



A recently developed linear algebraic method for the computation of perturbation expansion coefficients to large order is applied to the problem of a hydrogenic atom in a magnetic field. We take as the zeroth order approximation the $D \to \infty$ limit, where $D$ is the number of spatial dimensions. In this pseudoclassical limit, the wavefunction is localized at the minimum of an effective potential surface. A perturbation expansion, corresponding to harmonic oscillations about this minimum and higher order anharmonic correction terms, is then developed in inverse powers of $(D-1)$ about this limit, to 30th order. To demonstrate the implicit parallelism of this method, which is crucial if it is to be successfully applied to problems with many degrees of freedom, we describe and analyze a particular implementation on massively parallel Connection Machine systems (CM-2 and CM-5). After presenting performance results, we conclude with a discussion of the prospects for extending this method to larger systems.

## I. INTRODUCTION

The spatial dimension has long been treated as a variable parameter in analyzing critical phenomena and in other areas of physics.[1] However, only in the past ten years has this concept been extensively applied to atomic and molecular systems, particularly to develop dimensional scaling methods for electronic structure.[2] The motivation for this unconventional approach is that the Schrödinger equation reduces to easily solvable forms in the limits $D \to 1$ and/or $D \to \infty$. When both limiting solutions are available, interpolation in $1/D$ may be used to approximate the physically meaningful $D = 3$ result; this has yielded excellent results for correlation energies of two-electron atoms[3] and for $H_2$ Hartree-Fock energies.[4] Alternatively, if the $D \to \infty$ solutions are avaliable for both the problem of interest and a simpler model problem (e.g., Hartree-Fock) for which $D = 3$ results are easier to calculate, the latter may be used to "renormalize" some parameter (e.g., nuclear charge). Then the $D \to \infty$ solution with the renormalized parameter may give a good approximation to the $D = 3$ solution with the actual parameter value.[5]

Our work deals with another widely applicable dimensional scaling method, a perturbation expansion in inverse powers of $D$ or a related function, about the solution for the $D \to \infty$ limit. That limit is pseudoclassical and readily evaluated, as it reduces to the simple problem of minimizing an effective potential function.[2] For large but finite $D$, the first-order correction accounts for harmonic oscillations about this minimum, and higher-order terms provide anharmonic corrections. Dimensional perturbation theory has been applied quite successfully to the ground[6] and some excited states[7] two-electron atoms and to the hydrogen molecule-ion[8] using a "moment method" to solve the set of perturbation equations. However, this method is not easily extended to larger systems. It also requires a different program for each eigenstate, and does not directly provide an expansion for the wavefunction.

A recently developed linear algebraic method has overcome these shortcomings.[9] This is conceptually quite simple, so can easily be applied to systems with any number of degrees of freedom. It permits calculation of ground and excited energy levels using a single program and the wavefunction expansion coefficients are directly obtained in the course of computing the perturbation expansion for the energy. This method has thus far been applied to central force problems, including quasibound states for which the complex eigenenergy represents both the location and width of the resonance.[10]

The linear algebraic version of dimensional perturbation theory is also well suited to parallel computation. Here we demonstrate this for a prototype problem with two degrees of freedom, the hydrogen atom in a magnetic field. This system has received much attention; it exhibits chaotic behavior and poses difficulties that have challenged many theoretical techniques. Most theoretical approaches treat either the magnetic field or the Coulomb potential as a perturbation and therefore work best near either the low- or high-field limit, respectively. However, the leading terms of a perturbation expansion in inverse powers of $D$ include major portions of the nonseparable interactions in all field strengths. The efficacy of methods equivalent to the $1/D$ expansion has been demonstrated for the hydrogen atom in

a magnetic field[11] and for kindred problems with an electric field or crossed electric and magnetic fields,[12] although not in formulations suited to parallel computation. The present paper is devoted solely to implementing of the linear algebraic method on the Connection Machine, and to evaluating the performance of the computational algorithm as well as prospects for treating systems with more degrees of freedom. Numerical results for the ground and several excited states over a wide range of field strengths will be presented in a separate paper.[13]

## II. THEORY

The Schrödinger equation for a nonrelativistic hydrogenic atom in a uniform magnetic field along the $z$ axis, in atomic units and cylindrical coordinates, is given by

$$\left\{-\frac{1}{2}\left[\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}\right)+\frac{\partial^2}{\partial z^2}-\frac{m^2}{\rho^2}\right]+\frac{mB}{2}+\frac{B^2\rho^2}{8}-\frac{Z}{\sqrt{\rho^2+z^2}}\right\}\Psi(\rho,z)=E\Psi(\rho,z). \tag{1}$$

Here $m$ is the azimuthal quantum number and the magnetic field $B$ is measured in units $m_e^2e^3c/\hbar^3\simeq 2.35\times10^9$ G.[14] We can eliminate the first derivative term with the substitution

$$\Phi(\rho,z)=\rho^{1/2}\Psi(\rho,z), \tag{2}$$

which gives

$$\left\{-\frac{1}{2}\left(\frac{\partial^2}{\partial\rho^2}+\frac{\partial^2}{\partial z^2}\right)+\frac{m^2-\frac{1}{4}}{2\rho^2}+\frac{mB}{2}+\frac{B^2\rho^2}{8}-\frac{Z}{\sqrt{\rho^2+z^2}}\right\}\Phi(\rho,z)=E\Phi(\rho,z). \tag{3}$$

The two good quantum numbers are $m$ and the $z$ parity $\pi_z$, although the observation of exponentially small level anticrossings suggests a "hidden" approximate symmetry.[15] The trivial (i.e. diagonal in each $m_z^\pi$ subspace) $mB/2$ term may be dropped, to be added at the end of the calculation.

Now we consider the more general case where $\rho$ is the radius of a $(D-1)$-dimensional hypersphere (with $D-2$ remaining angles), so that the total number of spatial dimensions is $D$ when we include the component $z$ parallel to the magnetic field. In a manner completely analogous to that described above, we may eliminate the first derivative in the radial part of the Laplacian,

$$\frac{1}{\rho^{D-2}}\frac{\partial}{\partial\rho}\left(\rho^{D-2}\frac{\partial}{\partial\rho}\right),$$

by introducing the substitution

$$\Phi(\rho,z)=\rho^{(D-2)/2}\Psi(\rho,z), \tag{4}$$

which leads to the same form as Eq. (3), with $m^2-1/4$ replaced by $\Lambda(\Lambda+1)$ where $\Lambda=|m|+\frac{1}{2}(D-4)$.[16] As before, we drop the $mB/2$ term. Since the resulting equation depends on $m$ and $D$ only through $\Lambda$, we see that interdimensional degeneracies[16] connect states $(|m|,D)$ with $(|m|-1,D+2)$. It proves convenient to define a scaling parameter as $\kappa\equiv D+2|m|-1$ and take $\Lambda=\frac{1}{2}(\kappa-3)$. The entire $|m|=0,1,2,\ldots$ spectrum for the real three-dimensional system can then be recovered simply by obtaining the solutions for $\kappa=2,4,6,\ldots$, respectively.

On introducing dimensionally scaled variables defined by

$$\rho=\kappa^2\tilde{\rho},\qquad z=\kappa^2\tilde{z},\qquad \tilde{B}=\kappa^3B,\qquad \epsilon=\kappa^2E, \tag{5}$$

we obtain the dimension-dependent Schrödinger equation

$$\left\{-\frac{1}{2}\delta^2\left(\frac{\partial^2}{\partial\tilde{\rho}^2}+\frac{\partial^2}{\partial\tilde{z}^2}\right)+\frac{1-4\delta+3\delta^2}{8\tilde{\rho}^2}+\frac{\tilde{B}^2\tilde{\rho}^2}{8}-\frac{Z}{\sqrt{\tilde{\rho}^2+\tilde{z}^2}}\right\}\Phi(\tilde{\rho},\tilde{z})=\epsilon\Phi(\tilde{\rho},\tilde{z}), \tag{6}$$

where $\delta=1/\kappa$ is treated as a continuous perturbation parameter. The same form may be obtained simply by replacing $|m|$ in Eq. (3) by $(\kappa-2)/2$; this is the procedure employed by Bender and coworkers.[11]

We see that in the $\delta\to0$ limit, the problem reduces to finding the minimum of the effective potential

$$V_{\text{eff}}(\tilde{\rho},\tilde{z}) = \frac{1}{8\tilde{\rho}^2} + \frac{\tilde{B}^2\tilde{\rho}^2}{8} - \frac{Z}{\sqrt{\tilde{\rho}^2+\tilde{z}^2}}. \tag{7}$$

Clearly, the effective potential is minimized for $\tilde{z} = z_m = 0$, so we are left with the straightforward problem of minimizing a function of one variable to determine $\rho_m$ and $V_{\text{eff}}(\rho_m, z_m)$. In the zero-field limit, $\rho_m = (4Z)^{-1}$ and $V_{\text{eff}}(\rho_m, z_m) = -2Z^2$ in the scaled energy units of Eq. (5); in unscaled units this becomes

$$E = \frac{-Z^2}{2(|m|+1)^2}, \tag{8}$$

the correct expression for the ground state energy in each azimuthal manifold. In the strong-field limit, $\rho_m \approx \tilde{B}^{-1/2}$ and $V_{\text{eff}}(\rho_m, z_m) \approx \tilde{B}/4$, or in unscaled units

$$E \approx \frac{1}{2}B(|m|+1), \tag{9}$$

which is simply the continuum threshold in each $m^{\pi_z}$ subspace. This appropriate behavior in both limits, which has been noted previously,[11] is the motivation behind the definition of $\kappa$ which we have used.[17]

For large but finite $\kappa$ the system will undergo harmonic oscillations about this minimum, so we introduce dimension-scaled displacement coordinates $x_1$ and $x_2$ through

$$\tilde{\rho} = \rho_m + \delta^{1/2}x_1, \qquad \tilde{z} = \delta^{1/2}x_2. \tag{10}$$

Substituting into Eq. (6) leads to

$$\begin{aligned}\left\{-\frac{1}{2}\delta\left(\frac{\partial^2}{\partial x_1^2}+\frac{\partial^2}{\partial x_2^2}\right) + \delta\left(\frac{1}{2}\omega_1^2x_1^2+\frac{1}{2}\omega_2^2x_2^2+v_{0,0}\right) + V_{\text{eff}}(\rho_m,z_m) + \right.\\ \left.\delta\sum_{j=1}^{\infty}\delta^{j/2}\left[\left(\sum_{l=0}^{(j+2)/2} {}^{l}v_{j,j+2}x_1^{j+2-2l}x_2^{2l}\right) + v_{j,j}x_1^j + v_{j,j-2}x_1^{j-2}\right]\right\}\Phi(x_1,x_2) = \epsilon\Phi(x_1,x_2),\end{aligned} \tag{11}$$

where

$$\begin{aligned}\omega_1^2 &= \frac{3}{4\rho_m^4} - \frac{2Z}{\rho_m^3} + \frac{\tilde{B}^2}{4},\\ \omega_2^2 &= \frac{Z}{\rho_m^3},\\ v_{0,0} &= -\frac{3}{4\rho_m^2},\\ v_{1,-1} &= 0.\end{aligned}$$

We next expand the wavefunction and energy in Eq. (11) as

$$\epsilon = V_{\text{eff}}(\rho_m,z_m) + \delta\sum_{j=0}^{\infty}\epsilon_{2j}\delta^j, \tag{12}$$

$$\Phi(x_1,x_2) = \sum_{j=0}^{\infty}\Phi_j(x_1,x_2)\delta^{j/2} \tag{13}$$

and collect powers of $\delta^{1/2}$ to obtain an infinite set of inhomogeneous differential equations,

$$\sum_{j=0}^{p}(\mathcal{H}_j - \epsilon_j)\Phi_{p-j} = 0, \quad p = 0,1,2,\ldots, \tag{14}$$

where

$$\mathcal{H}_0 = -\frac{1}{2}\left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}\right) + \frac{1}{2}\omega_1^2 x_1^2 + \frac{1}{2}\omega_2^2 x_2^2 + v_{0,0} \tag{15a}$$

$$\mathcal{H}_{j>0} = \left(\sum_{l=0}^{(j+2)/2} {}^{l}v_{j,j+2} x_1^{j+2-2l} x_2^{2l}\right) + v_{j,j} x_1^j + v_{j,j-2} x_1^{j-2} \tag{15b}$$

with $\epsilon_{2i+1} = 0$. The case $p = 0$ corresponds to a pair of independent harmonic oscillators, with the solution

$$\epsilon_0 = \left(\nu_1 + \tfrac{1}{2}\right)\omega_1 + \left(\nu_2 + \tfrac{1}{2}\right)\omega_2 + v_{0,0}, \tag{16}$$

$$\Phi_0(x_1, x_2) = h_{\nu_1}(\omega_1^{1/2} x_1) h_{\nu_2}(\omega_2^{1/2} x_2), \tag{17}$$

where $\nu_1$ and $\nu_2$ are quantum numbers for the normal modes corresponding to motion perpendicular to and parallel to the magnetic field, respectively, and the $h_i$ are the one-dimensional harmonic oscillator eigenfunctions. The correspondences between these large-dimension quantum numbers and the conventional labels for the low, intermediate, and high field cases will be detailed elsewhere.[13]

## III. TENSOR METHOD

The higher-order terms in the energy and wavefunction expansions may be computed using the linear algebraic method.[9] The wavefunction expansion terms $\Phi_j(x_1, x_2)$ are expanded in terms of the harmonic oscillator eigenfunctions $h_i$,

$$\Phi_j(x_1, x_2) = \sum_{i_1}\sum_{i_2} {}^{i_1,i_2}a_j h_{i_1}(\omega_1^{1/2} x_1) h_{i_2}(\omega_2^{1/2} x_2), \tag{18}$$

so that the representation $\mathbf{a}_j$ is a tensor of rank 2. The displacement coordinates $x_i$ are represented in this basis by the matrix

$$\mathbf{x}_i = \frac{1}{\sqrt{2\omega_i}} \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & \\ \sqrt{1} & 0 & \sqrt{2} & 0 & \\ 0 & \sqrt{2} & 0 & \sqrt{3} & \cdots \\ 0 & 0 & \sqrt{3} & 0 & \\ & & \vdots & & \ddots \end{pmatrix}, \tag{19}$$

and the Hamiltonian $\mathcal{H}_{j>0}$ is represented as a linear combination of direct (outer) products of these matrices, namely

$$\mathbf{H}_{j>0} = \left(\sum_{l=0}^{(j+2)/2} {}^{l}v_{j,j+2} \mathbf{x}_1^{j+2-2l} \otimes \mathbf{x}_2^{2l}\right) + v_{j,j}\mathbf{x}_1^j \otimes \mathbf{I} + v_{j,j-2}\mathbf{x}_1^{j-2} \otimes \mathbf{I}. \tag{20}$$

Using these representations, the $j = 0$ term of Eq. (14), $(\mathcal{H}_0 - \epsilon_0)\Phi_p$, becomes

$$(\mathbf{H}_0 - \epsilon_0\mathbf{I})\mathbf{a}_p = ((\omega_1\mathbf{K}_1) \oplus (\omega_2\mathbf{K}_2))\mathbf{a}_p, \tag{21}$$

where $\mathbf{K}_i$ is a diagonal matrix with element $(j, j)$ equal to $j - \nu_i$, since the basis functions $h_{i_1}(\omega_1^{1/2} x_1) h_{i_2}(\omega_2^{1/2} x_2)$ are eigenfunctions of $\mathcal{H}_0$. Let us pause at this point to be sure that there is no confusion regarding the notation. A "direct sum" is analogous to the direct product operation, i.e. $\mathbf{C} = \mathbf{A} \oplus \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are matrices, gives a rank 4 tensor with elements $C_{j_1k_1j_2k_2} = A_{j_1k_1} + B_{j_2k_2}$. Then the multiplication $\mathbf{E} = \mathbf{CD}$, where $\mathbf{D}$ and $\mathbf{E}$ are rank 2 tensors, implies $E_{j_1j_2} = C_{j_1k_1j_2k_2}D_{k_1k_2}$, where we are using the implied summation convention. If we define a rank 4 tensor $\widetilde{\mathbf{K}}$ with elements

$$\widetilde{\mathbf{K}}_{j_1k_1j_2k_2} = \begin{cases} 1, & j_1 = k_1 = \nu_1, j_2 = k_2 = \nu_2 \\ (\omega_1(j_1 - \nu_1) + \omega_2(j_2 - \nu_2))^{-1}, & j_1 = k_1, j_2 = k_2, j_1 \neq \nu_1 \text{ and/or } j_2 \neq \nu_2 \\ 0, & \text{otherwise} \end{cases}, \tag{22}$$

then it may be verified that $\widetilde{\mathbf{K}}((\omega_1\mathbf{K}_1) \oplus (\omega_2\mathbf{K}_2))$ is equal to the direct product $\mathbf{I} \otimes \mathbf{I}$ except for a zero in element $\nu_1\nu_1\nu_2\nu_2$. However, due to the orthogonality condition

$$\mathbf{a}_0^T \mathbf{a}_p = \delta_{0,p}, \tag{23}$$

where $\delta_{0,p}$ is the Kronecker delta, we have

$$\widetilde{\mathbf{K}}((\omega_1 \mathbf{K}_1) \oplus (\omega_2 \mathbf{K}_2))\mathbf{a}_p = \mathbf{a}_p, \qquad p > 0. \tag{24}$$

Therefore if we multiply Eq. (14) on the left by $\widetilde{\mathbf{K}}$, we obtain a recursive solution for $\mathbf{a}_p$,

$$\mathbf{a}_p = -\widetilde{\mathbf{K}} \sum_{j=1}^{p} (\mathbf{H}_j - \epsilon_j \mathbf{I})\mathbf{a}_{p-j}. \tag{25}$$

If we instead multiply Eq. (14) on the left by $\mathbf{a}_0^T$ and use the orthogonality condition of Eq. (23), we obtain an expression for $\epsilon_p$ in terms of $\mathbf{a}_{j<p}$,

$$\epsilon_p = \sum_{j=1}^{p} \mathbf{a}_0^T \mathbf{H}_j \mathbf{a}_{p-j}. \tag{26}$$

From a computational standpoint, the only operation with which we need to concern ourselves is the multiplication of a rank 4 tensor with a rank 2 tensor, $\mathbf{H}_j \mathbf{a}_{p-j}$. According to Eq. (22), the nonzero elements of $\widetilde{\mathbf{K}}$ appear in Eq. (25) simply as scaling factors for each element of $\mathbf{a}_p$ after the summation has been completed.

## IV. IMPLEMENTATION

### A. Rewriting the recursion relations

In order to solve Eqs. (25) and (26) recursively, we introduce the rank 2 tensor $\boldsymbol{\mathcal{A}}_{jln}$ which is defined as

$$\boldsymbol{\mathcal{A}}_{jln} \equiv (\mathbf{x}_1^{j-2l} \otimes \mathbf{x}_2^{2l})\mathbf{a}_n, \tag{27}$$

so that the $\mathbf{H}_j \mathbf{a}_{p-j}$ term which most concerns us may be expressed as a linear combination of these tensors, namely

$$\mathbf{H}_j \mathbf{a}_{p-j} = \left( \sum_{l=0}^{(j+2)/2} {}^{l}v_{j,j+2} \boldsymbol{\mathcal{A}}_{j+2,l,p-j} \right) + v_{j,j} \boldsymbol{\mathcal{A}}_{j,0,p-j} + v_{j,j-2} \boldsymbol{\mathcal{A}}_{j-2,0,p-j}. \tag{28}$$

In order for this change of notation to be beneficial, we need to find some simple recursion relation(s) relating the $\boldsymbol{\mathcal{A}}_{jln}$ tensors for a given order to those tensors which have been used at lower orders. In addition, we hope that only a small subset of the tensors used for previous orders are needed, so that a managable number of these tensors (and the wavefunction tensors $\mathbf{a}_n$) need to be allocated in memory. Two recursion relations are immediately evident, namely

$$\boldsymbol{\mathcal{A}}_{j+1,l,n} = (\mathbf{x}_1 \otimes \mathbf{I})\boldsymbol{\mathcal{A}}_{jln}, \tag{29a}$$

$$\boldsymbol{\mathcal{A}}_{j+2,l+1,n} = (\mathbf{I} \otimes \mathbf{x}_2^2)\boldsymbol{\mathcal{A}}_{jln}. \tag{29b}$$

It remains to be seen how many terms may be computed using these relations, and conversely how many will need to be computed "from scratch." We show in Fig. 1 the tensors $\boldsymbol{\mathcal{A}}_{jln}$ which appear in the first term of Eq. (28) for the first few orders $p$, and also indicate which recursion relations (if any) may be used to compute each tensor (with preference given to Eq. (29a) if both are applicable). We see that Eq. (29a) simply "shifts" the staircase-like pattern of tensors to the right, while Eq. (29b) fills in all but three of the gaps left by this shift. Turning our attention to the second term of Eq. (28), we note that at order $p$ this term represents tensors which have appeared in the first term at order $p-2$, with the exception of $\boldsymbol{\mathcal{A}}_{1,0,p-1}$ and $\boldsymbol{\mathcal{A}}_{2,0,p-2}$. Similarly, the third term of Eq. (28) at order $p$ involves tensors which occured in the second term at order $p-2$, with the exception of $\boldsymbol{\mathcal{A}}_{0,0,p-2}$, which is simply the wavefunction tensor $\mathbf{a}_{p-2}$ which is resident in memory. Therefore, at each order we have at most five tensors which cannot be computed from direct application of Eqs. (29).

In reality, the computation of these five tensors is no more difficult than the two standard recursion relations:

$$
\begin{aligned}
\boldsymbol{\mathcal{A}}_{1,0,p-1} &= (\mathbf{x}_1 \otimes \mathbf{I})\mathbf{a}_{p-1}, \\
\boldsymbol{\mathcal{A}}_{3,0,p-1} &= (\mathbf{x}_1^2 \otimes \mathbf{I})\boldsymbol{\mathcal{A}}_{1,0,p-1}, \\
\boldsymbol{\mathcal{A}}_{3,1,p-1} &= (\mathbf{I} \otimes \mathbf{x}_2^2)\boldsymbol{\mathcal{A}}_{1,0,p-1}, \\
\boldsymbol{\mathcal{A}}_{2,0,p-2} &= (\mathbf{x}_1^2 \otimes \mathbf{I})\mathbf{a}_{p-2}, \\
\boldsymbol{\mathcal{A}}_{4,2,p-2} &= (\mathbf{I} \otimes \mathbf{x}_2^2)[(\mathbf{I} \otimes \mathbf{x}_2^2)\mathbf{a}_{p-2}].
\end{aligned}
\tag{30}
$$

We note that these relations, as well as the two standard recursion relations, only involve computation along one coordinate axis.

By properly ordering the computation steps, we may compute all of the tensors $\boldsymbol{\mathcal{A}}_{jln}$ at order $p$ "in place" (i.e. overwriting all of the tensors from order $p-1$), thus minimizing the need for temporary storage. The second recursion relation uses tensors from order $p-2$, which must be temporarily stored during the order $p-1$ calculation. However, this relation is only used for $(p-2)/2$ tensors at order $p$, so the additional storage cost is minor. The algorithm may be summarized as follows (see also Fig. 2): (1) Recursion Eq. (29a) is applied to all tensors, working from right to left in Fig. 2 to avoid the need for temporary storage. Note that this step does *not* overwrite those tensors which we will need later for Eq. (29b). (2) Recursion Eq. (29b) is applied to tensors from order $p-2$ which are stored separately. By again working from right to left, we may replace the tensors in temporary storage as they are used with the corresponding tensors from order $p-1$, so that both the main set of tensors and the temporary set are updated as they are used. (3) Five original tensors (Eq. (30)) are computed. (4) $\mathbf{a}_p$ is computed by taking a linear combination of tensors, and the contributions of those tensors which are necessary for $\mathbf{a}_{p+2}$ and $\mathbf{a}_{p+4}$ are now computed.

### B. Implementing the recursion relations on the Connection Machine

We have implemented the computation of the energy eigenvalues, as described in the previous section, using a data-parallel algorithm on the CM-5 Connection Machine computer. The CM-5 computer consists of up to 16,384 processor nodes, each consisting of a Sun Sparc processor and four vector units for fast floating-point computation. It supports both the data-parallel and the message-passing styles of parallel computation.

The data-parallel capabilities of the CM-5 are supported in the form of high-level languages: C* is a data-parallel extension of the C programming language; while CM Fortran is similar to the Fortran 90 draft standard, augmented by some features from High-Performance Fortran (HPF) such as the `FORALL` statement. For this work, we chose to use CM Fortran.

The basic parallel data type in CM Fortran is the usual Fortran array. When the CM Fortran compiler encounters a `DIMENSION` statement, it allocates memory to spread the array across the processors of the CM-5. If it is desired that certain axes of a multidimensional array be localized to a single processor, that can be arranged by a simple compiler directive, called the `LAYOUT` directive. Axes localized in this way are called *serial axes*, while those spread across the processor array are called *news axes*.

Operations on corresponding elements of arrays with identical layouts can then be performed by each processor in parallel. If there are more array elements than physical processors, the compiler arranges for each physical processor to contain a *subgrid* of multiple array elements, and array operations are then multiplexed over these elements as required. The compiler's orchestration of this process is completely transparent to the CM Fortran user, who may then regard each array element as living in its own *virtual processor*.

In this picture, operations involving noncorresponding array elements require interprocessor communication. Data-parallel programming languages, such as CM Fortran, support many different varieties of this. For the problem at hand, however, only *nearest-neighbor communication* is used.

Nearest-neighbor communication is needed when, for example, you would like to take a linear combination of the array elements in (virtual) processors, $i-1$, $i$, and $i+1$. It is supported by a Fortran 90 (and CM Fortran) intrinsic function known as `CSHIFT`. If `A` is a CM Fortran array, and `i` and `j` are integers, then `CSHIFT(A,i,j)` is another such array, whose elements are shifted along axis `i` by an amount `j`. There is another variant of this function, known as `EOSHIFT`, that does basically the same thing, differing only in its treatment of the boundary elements of the array; whereas `CSHIFT` treats the boundaries as though the array were periodic, `EOSHIFT` has extra arguments for arrays of codimension one to be inserted at the boundary.

Our storage method uses the fact that the position matrices in the harmonic oscillator representation, given by Eq. (19), are doubly banded. We store only the nonzero elements of the $i$th row in (virtual) processor $i$. Thus, one can think of the band below the diagonal as a one-dimensional array, `XL`, and that above the diagonal as another one-dimensional array, `XU`. Then, `XL(i)` and `XU(i)` contain the two nonzero elements in row `i`, and we can write

$$
\mathbf{x}_{\alpha\beta} = \texttt{XL}(\alpha)\delta_{\alpha-1,\beta} + \texttt{XU}(\alpha)\delta_{\alpha+1,\beta}.
$$

The main operation needed for the implementation of the recursion relations, Eqs. (29), is the inner product of the position matrices with the matrices, $\boldsymbol{\mathcal{A}}_{j,l,n}$. The latter are stored as dense three-dimensional arrays, with two *news* axes for the components of the matrices, and one *serial* axis representing the allowed combinations of $j$, $l$, and $n$. Thus, one can imagine these matrices as spread across the processor array, with corresponding components localized to the same (virtual) processor.

Consider one of these matrices; call it $\boldsymbol{\mathcal{A}}_{\mu\nu}$ (where the Greek indices now label the *tensor* components and *not* the serial axis, $\{j,l,n\}$, which is being supressed for the moment). Then Eq. (29a), which is the inner product of $\mathbf{x}$ with (the first component of) $\boldsymbol{\mathcal{A}}$, is

$$(\mathbf{x}_{\xi\mu} \otimes \mathbf{I})\boldsymbol{\mathcal{A}}_{\mu\nu} = \texttt{XL}(\xi)\boldsymbol{\mathcal{A}}_{\xi-1,\nu} + \texttt{XU}(\xi)\boldsymbol{\mathcal{A}}_{\xi+1,\nu}. \tag{31}$$

Note that we can implement this in terms of two `CSHIFT` operations along the first axis of $\boldsymbol{\mathcal{A}}$. Since this must be done for all values of $\mu$, we add this *instance axis* to the arrays, `XL` and `XU`.

The complete algorithm for implementing the inner product then begins by dimensioning the arrays as follows:

```
INTEGER m,n
DIMENSION XL(nmax,nmax), XU(nmax,nmax), AA(nmax,nmax)
```

The `XL` and `XU` arrays are then initialized as follows:

```
FORALL(m=1:nmax,n=1:nmax) XL(m,n) = SQRT(n-1)
FORALL(m=1:nmax,n=1:nmax) XU(m,n) = SQRT(n)
```

where the `FORALL` statements are parallel `DO`-loop constructions supported in High-Performance Fortran (and CM Fortran). For example, the first such statements cause each (virtual) processor to take its own index, decrement it by one, and take the square root. Alternatively, rather than allocate a separate array for `XU`, we can obtain it from `XL` by a simple `EOSHIFT` operation.

The inner product in Eq. (31) is then taken as follows:

```
XA = XL*EOSHIFT(AA,1,-1) + XU*EOSHIFT(AA,1,+1)
```

Using this method, the recursion relations, Eqs. (29) and (30), can be implemented, and the terms in Eq. (28) can be multiplied by the scalars, $v_{i,j}$, and summed in place. (Since the $\{j,l,n\}$ axis is serial, the arrays $\boldsymbol{\mathcal{A}}_{j,l,n}$ are all aligned in the processor array.) Similarly, the sum over $j$ in Eq. (26) can be taken (without the premultiplication by $\mathbf{a}_0$, which is independent of $j$.)

Finally, the result for $\epsilon_p$, given by Eq. (26), can be found quite easily. Since $\mathbf{a}_0$ is an eigenvector in our representation, we can effectively premultiply it by simply taking the first component of $\sum_j \mathbf{H}_j \mathbf{a}_{p-j}$. The statement

```
ep = XA(1,1)
```

reaches into the (virtual) processor containing the `(1,1)` component of the array `XA`, pulls out its value, and writes it into the scalar (stored on the control processor) quantity `ep`, from where it can be manipulated, output, etc.

Thus, the array extensions of Fortran 90 and High-Performance Fortran, as embodied in the CM Fortran language, give a convenient set of primitives for the efficient implementation of the dimensional perturbation theory algorithm.

### C. Contraction of axis lengths

All of the wavefunction tensors $\mathbf{a}_n$ and the tensors $\boldsymbol{\mathcal{A}}_{jln}$ used in their computation are allocated to the size of the final wavefunction tensor, $\mathbf{a}_{p_{max}}$, which permits calculation of energy expansion coefficients through $\epsilon_{p_{max}+1}$. However, we have not yet attempted to take advantage of the sparseness of these tensors. To illustrate the amount of room for improvement, Fig. 3 indicates the nonzero elements of the first few wavefunction tensors for the case of a ground state ($\nu_1 = \nu_2 = 0$) calculation. Since we would lose much of the efficiency of the routines for the recursion relations described above if we attempted to treat these as general sparse tensors, we will instead look for smaller-scale improvements.

First, we see that since $\mathbf{x}_2$ only occurs in even powers, alternating columns are always zero and need not be stored. In addition to cutting the storage in half, this reduces the two next-nearest neighbor communications required for the $(\mathbf{I} \otimes \mathbf{x}_2^2)$ multiplication to two nearest neighbor communications.

Looking along the other axis, we see that either even or odd rows, but not both in a given tensor, contain nonzero elements, so that the rows may be paired up as long as we maintain some sort of flag telling us whether the even or odd row of the pair is represented. This also carries an extra communication benefit in addition to the storage improvement; one of the two nearest-neighbor communications in the $(\mathbf{x}_1 \otimes \mathbf{I})$ multiplication is converted to an entirely local operation.

## V. PERFORMANCE

We now turn to the mapping of these arrays to the individual processors on the Connection Machine, in the typical case where the array size exceeds the number of processors. The recursion relation of Eq. (29a) may be applied independently to different columns since the communication is entirely within columns. Similarly, Eq. (29b) may operate independently on separate rows. Since the former recursion relation is applied much more often than the latter, we would expect that the optimal array layout would be that in which each processor operates on a small number of long column segments, i.e. the first (row) axis is "more serial" while the second (column) axis is "more parallel."

We can in fact obtain a quantitative measure of this balance. Suppose we have an $N \times N$ matrix to be allocated on a system of $M$ processors. If we assign $n$ columns (or segments of columns) to each processor, then the length of each of these column segments is $N^2/nM$, as shown in Fig. 4. The first recursion relation may be implemented in CM Fortran as described in section IV. B, with the addition of an `IF` clause due to the contraction of the first axis, as described in the previous section. Using appropriately defined arrays `X0` and `XLU`, this may be accomplished as follows:

```
IF (odd rows nonzero) THEN
   XA = X0*AA + XLU*EOSHIFT(AA,1,-1)
ELSE
   XA = X0*AA + EOSHIFT(XLU*AA,1,+1)
ENDIF
```

Whichever branch of this clause is taken, there are three arithmetic operations on each array element, with a total cost of $3N^2 t_A/M$, where $t_A$ represents the time for a single arithmetic operation. The `EOSHIFT` operation will move $n$ elements from any given processor into an adjacent processor, while the remaining $N^2/M - n$ elements require an on-processor move (except for the case $n = N/M$, where all elements are moved on-processor). Thus the `EOSHIFT` time is given by

$$\begin{array}{ll} nt_Q & \text{if } n = N/M \\ nt_Q + \left(\frac{N^2}{M} - n\right) t_M & \text{otherwise} \end{array},$$

where $t_Q$ represents the queue waiting time for interprocessor communication and $t_M$ is the on-processor move time.

The second recursion relation can be effected by the CM Fortran statement

```
XA = X20*AA + EOSHIFT(X2LU*AA,2,-1) + X2LU*EOSHIFT(AA,2,+1)
```

where the definition of the constant arrays `X20` and `X2LU` is dependent on the contraction of the second axis, as described in the previous section. The arithmetic cost is $5N^2 M t_A/M$ and the cost of the two `EOSHIFT` operations is

$$\begin{array}{ll} \frac{2N^2}{nM} t_Q & \text{if } n = N \\ \frac{2N^2}{nM} t_Q + 2\left(\frac{N^2}{M} - \frac{N^2}{nM}\right) t_M & \text{otherwise} \end{array}.$$

For a program which applies the first recursion relation $n_1$ times and the second relation $n_2$ times, the total time incurred by the two relations is

$$t_{recur} = \frac{N^2}{M}[(3n_1 + 5n_2)t_A + (n_1 + 2n_2)t_M] + \left(n_1 n + \frac{2n_2 N^2}{nM}\right)(t_Q - t_M), \tag{32}$$

where we have neglected the two special cases for $n$ described above. The first term is independent of the specific layout, described by the parameter $n$. Since the only interprocessor data motion in the entire program arises from the recursion relations, the second term can be used to determine the optimal array layout. Since $t_Q > t_M$, we want to minimize the term $n_1 n + 2n_2 N^2/nM$. Setting its derivative with respect to $n$ equal to zero gives a simple expression for the optimal layout parameter $n_{opt}$,

$$n_{opt} = \sqrt{\frac{2n_2 N^2}{n_1 M}}. \tag{33}$$

For example, a 30th order calculation using the implementation discussed here requires $N = 128$, $n_1 = 18441$, $n_2 = 925$. A 64-processor CM-5 contains 256 vector units (hence $M = 256$). Thus, we would estimate $n_{opt} \approx 2.5$,

in agreement with the empirical observation that $n = 2$ results in faster run times than either $n = 1$ or $n = 4$ on a 64-processor CM-5 (see Table I). Special care must be paid to the special case $n = N/M$ when $M \leq N$, for a layout with the first axis entirely serial ($n = 1$) may be optimal despite the fact that Eq. (33) gives a larger value for $n_{opt}$.

Table I presents timings and the corresponding floating point rates for 20th and 30th order calculations. The recursion relations involve substantial interprocessor communication; for the problem sizes considered here, we have found that approximately 40% of the CM execution time is spent performing arithmetic, the remainder being consumed by communication operations.

In spite of this, we were able to obtain a performance of nearly 20 Mflops per CM-5 processor node for the largest subgrid sizes considered here.[18] Since the algorithm described here is fully scalable, we expect that if the subgrid size was kept fixed and the problem was ported to the largest CM-5's existing today (1024 processor nodes), the total performance would be about 20 Gflops.

Keeping the subgrid size fixed, however, implies that the problem size grows with the hardware. Again referring to Table I, we see that if we move a fixed-size problem to larger machines, the per-processor performance degrades. This is because the correspondingly decreasing subgrid size means that a larger fraction of time is being spent on latencies (overhead costs).

Thus, massively parallel implementations of this algorithm will be useful if there is something to be gained by going to larger problem sizes. The most obvious way to do this is to increase the order of the perturbation series, and hence the sizes of the arrays involved, the number of recursion relations that must be used, and the accuracy of the results. Here, however, we run into a problem. Table II shows that at 30th order we have exhausted (or nearly exhausted) the significant digits of the IEEE standard double-precision floating-point numbers used in the calculation. Thus, to grow the problem in this direction, we would need to employ quadruple-precision arithmetic.

There are, however, other options for growing the problem size in a useful manner. Table III shows results for several different magnetic field strengths. Indeed, one of the more interesting things to study in chaotic systems of this sort is the continuous dependence of the energy levels on the magnetic field strength, or other system parameter. Variation of these parameters provides another dimension over which to parallelize, and thereby maintain large subgrid sizes.

## VI. CONCLUSIONS

In Table II we give ground state expansion coefficients $\epsilon_{2j}$ for two field strengths, $B = 1$ and 1000, computed in double-precision arithmetic. Comparison with quadruple-precision calculations shows that the accumulation of roundoff error causes a linearly decreasing number of significant digits in the coefficients, most pronounced for those series with the slowest rate of growth in the coefficients.

In their work, Bender, Mlodinow, and Papanicolaou applied Shanks extrapolation to accelerate convergence.[11] For the higher-order series we have computed, the large order divergent behavior due to singularities renders this method ineffective.[19] Consequently, we employ Padé approximants to sum our series. Table III compares our $1/\kappa$ results for the ground state of the $m = 0$ and $-1$ mainfolds with variational calculations[20] and lower and upper bounds,[21] where available. As noted by Goodson and Watson, the order at which roundoff error leaves no significant digits in the expansion coefficient $\epsilon_{2j}$ may be determined by noting qualitative changes in the root and ratio tests.[7] We also give in Table III these maximum orders which are attainable using double-precision arithmetic. Results for excited states will be presented elsewhere.[13]

For general problems with $\tau$ degrees of freedom, the $\mathbf{a}_n$ and $\boldsymbol{\mathcal{A}}_n$ objects are tensors of rank $\tau$. It is clear that storing all of the tensor elements will rapidly become impractical; while a $128 \times 128$ array of double precision numbers consumes 131 kBytes of memory, a $128 \times 128 \times 128 \times 128$ object requires 2 GB. Even using a parallel I/O device such as the DataVault or Scalable Disk Array for temporary storage, it is evident that it will become necessary to incorporate the sparse nature of these data structures more explicitly and to reduce the communication (especially to external devices) at the expense of additional arithmetic wherever possible.

It has recently been shown[9] that the recursion relations may be reordered so that only the $\mathbf{a}_n$ tensors need to be stored. This approach has been used in the computation of expansion coefficients for the helium atom, a system with three degrees of freedom. Numerical estimates of the storage and computational costs associated with this approach indicate that large order calculations (20th order or higher) are at present restricted to at most six degrees of freedom, *e.g.* a three-electron atom.[9] However, a promising approach may be to combine low-order perturbation expansions with other methods. In particular, the recent development of expansions about the $D \to 0$ limit[22] may provide means to augment the $D \to \infty$ results. Expansions about both of these limits can be enhanced by renormalization schemes.

## Acknowledgements

T.C.G. gratefully acknowledges the award of a Computational Science Graduate Fellowship from the U.S. Department of Energy, and would like to thank Thinking Machines Corporation for their hospitality during the time this work was performed.

---

[1] L.G. Yaffe, Rev. Mod. Phys. **54**, 407 (1982).
[2] *Dimensional Scaling in Chemical Physics*, edited by D.R. Herschbach, J. Avery, and O. Goscinski (Kluwer, Dordrecht, 1992).
[3] J.G. Loeser and D.R. Herschbach, J. Phys. Chem. **89**, 3444 (1985); J.G. Loeser and D.R. Herschbach, J. Chem. Phys. **86**, 3512 (1987).
[4] M. López-Cabrera, A.L. Tan, and J.G. Loeser, J. Phys. Chem. **97**, 2467 (1993).
[5] S. Kais, S.M. Sung, and D. R. Herschbach, J. Chem. Phys. **99**, 5184 (1993).
[6] D.Z. Goodson and D.R. Herschbach, Phys. Rev. Lett. **58**, 1628 (1987); D.Z. Goodson *et al.*, J. Chem. Phys. **97**, 8481 (1992).
[7] D.Z. Goodson and D.K. Watson, Phys. Rev. A **48**, 2668 (1993); see also Ref. 2, Section 8.1.
[8] M. López-Cabrera *et al.*, Phys. Rev. Lett. **68**, 1992 (1992).
[9] M. Dunn *et al.*, manuscript in preparation.
[10] T.C. Germann and S. Kais, J. Chem. Phys. **99**, 0000 (1993).
[11] C.M. Bender, L.D. Mlodinow, and N. Papanicolaou, Phys. Rev. **A 25**, 1305 (1982).
[12] V.S. Popov *et al.*, Phys. Lett. **124A**, 77 (1987); V.S. Popov *et al.*, Phys. Lett. **149A**, 418 (1990); V.S. Popov, V.D. Mur, and A.V. Sergeev, Phys. Lett. **149A**, 425 (1990). See also Ref. 2, Section 6.2.
[13] T.C. Germann *et al.*, work in progress.
[14] H. Friedrich, *Theoretical Atomic Physics* (Springer-Verlag, Berlin, 1990), Section 3.4.2.
[15] D.R. Herrick, Phys. Rev. A **26**, 323 (1982).
[16] D.R. Herrick, J. Math. Phys. **16**, 281 (1975); Ref. 2, Chapter 2.
[17] U. Sukhatme and T. Imbo, Phys. Rev. D **28**, 418 (1983) have advocated general shifted expansions such as $\kappa \equiv D + 2|m| - a$.
[18] No herculean programming effort was used to obtain this performance. The entire code was written in standard CM Fortran. It is possible that the use of utility library routines, scientific software library routines, or lower-level code might further improve this performance, but this has not been investigated.
[19] C.M. Bender and S.A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers* (McGraw-Hill, New York, 1978).
[20] W. Rösner *et al.*, J. Phys. B **17**, 29 (1984).
[21] C.R. Handy *et al.*, Phys. Rev. Lett. **60**, 253 (1988); **62**, 2199 (1989); see also comment by R.C. Rech, M.R. Gallas, and J.A.C. Gallas, Phys. Rev. Lett. **62**, 2198 (1989).
[22] C.M. Bender, S. Boettcher, and L. Lipatov, Phys. Rev. Lett. **68**, 3674 (1992).

| Architecture | $P$ | $n$ | CM time (sec) | MFLOPS/sec/node |
|---|---|---|---|---|
| | | 20TH ORDER CALCULATION (N=64) | | |
| CM-5 (VU) | 32 | 2 | 0.896 | 5.012 |
| | 128 | 2 | 0.730 | 1.536 |
| | | 30TH ORDER CALCULATION (N=128) | | |
| CM-2 | 256 | 1 | 11.91 | 0.588 |
| | 512 | 1 | 7.11 | 0.492 |
| CM-5 (VU) | 4 | 1 | 22.78 | 19.668 |
| | 32 | 1 | 4.62 | 12.124 |
| | 64 | 1 | 3.81 | |
| | 64 | 2 | 3.68 | 7.608 |
| | 64 | 4 | 3.75 | |
| | 128 | 2 | 2.82 | 4.964 |

TABLE I. Performance results on Connection Machines with $P$ nodes, where the optimal array layout is described by $n$ (see text). For the CM-2 using the slicewise execution model, each floating-point processor node is comprised of 32 bit-serial processors and a floating-point accelerator chip, so a full 64K CM-2 has 2048 processor nodes ($M = P$). For the CM-5, each processing node (PN) has four vector units, so $M = 4P$ in predicting performance.

| $j$ | $\epsilon_{2j}$ $(B=1)$ | $\epsilon_{2j}$ $(B=1000)$ |
|---|---|---|
| $-1$ | $-1.577218587578393$ | $1.910051627706109(3)$ |
| 0 | $0.63293278553615_1$ | $3.61765949346725_3(2)$ |
| 1 | $-0.3281655376303_1$ | $-1.61755795476809_6(3)$ |
| 2 | $0.189180754067_1$ | $8.71028965022799_1(3)$ |
| 3 | $-0.1202775104_2$ | $-5.8564694134795_7(4)$ |
| 4 | $0.10221091_4$ | $4.518169171696_0(5)$ |
| 5 | $-0.1685588_6$ | $-2.55253312427_6(6)$ |
| 6 | $0.46534_7$ | $-4.75598614937_9(7)$ |
| 7 | $-1.486_3$ | $3.36811883775_9(9)$ |
| 8 | $4.92_6$ | $-1.4285156162_5(11)$ |
| 9 | $-1.6_8(1)$ | $5.3728029857_7(12)$ |
| 10 | $1._0(2)$ | $-1.876673877_8(14)$ |
| 11 | | $5.834807058_2(15)$ |
| 12 | | $-1.29765909_0(17)$ |
| 13 | | $-1.39749835_0(18)$ |
| 14 | | $4.7606391_8(20)$ |
| 15 | | $-4.5262663_0(22)$ |
| 16 | | $3.365746_4(24)$ |
| 17 | | $-2.179148_6(26)$ |
| 18 | | $1.21980_8(28)$ |
| 19 | | $-5.17984_0(29)$ |
| 20 | | $3.7664_0(30)$ |
| 21 | | $2.6805_1(33)$ |
| 22 | | $-4.395_8(35)$ |
| 23 | | $5.073_1(37)$ |
| 24 | | $-4.92_4(39)$ |
| 25 | | $4.09_7(41)$ |
| 26 | | $-2.6_8(43)$ |
| 27 | | $7.8_6(44)$ |
| 28 | | $1._5(47)$ |
| 29 | | $-4._0(49)$ |

TABLE II. Expansion coefficients $\epsilon_{2j}$ for the ground-state energy, defined in Eq.(12), for field strengths $B = 1$ and 1000. Note that due to the scaling of Eq.(5), $\epsilon$ should be divided by $(D-1)^2 = 4$ to give the $m = 0$ energy $E$. Subscripts specify the digit after the last digit which is expected to be significant, and the number in parenthesis following each entry is the power of 10 multiplying that entry.

| $B$ | order | Present work | Rösner *et al.*[20] | Handy *et al.*[21] | | |
|---|---|---|---|---|---|---|
| | | | $E_B$, $1s_0$ state | | | |
| 0.1 | 7 | 0.54752648 | 0.547527 | | | |
| 1 | 11 | 0.831169 | 0.831169 | | | |
| 2 | 12 | 1.022213 | 1.022214 | 1.0222138 | $< E_B <$ | 1.0222142 |
| 20 | 20 | 2.21539 | 2.215399 | 2.215325 | $< E_B <$ | 2.215450 |
| 200 | 24 | 4.7275 | 4.727 | 4.710 | $< E_B <$ | 4.740 |
| 300 | 25 | 5.362 | 5.361 | 5.34 | $< E_B <$ | 5.39 |
| 1000 | 30 | 7.67 | 7.662 | 7.55 | $< E_B <$ | 7.85 |
| | | | $E_B$, $2p_{-1}$ state | | | |
| 0.1 | 10 | 0.20084567 | 0.2008456 | | | |
| 1 | 16 | 0.4565971 | 0.456597 | | | |
| 2 | 19 | 0.599613 | 0.599613 | | | |
| 20 | 25 | 1.46551 | 1.4655 | | | |
| 200 | 33 | 3.3473 | 3.3471 | | | |
| 300 | 35 | 3.83485 | 3.8346 | | | |
| 1000 | 38 | 5.640 | 5.63842 | | | |

TABLE III. Comparison of binding energies, $E_B = B(|m| + 1)/2 - E$, of the lowest energy states of the $m = 0$ and $-1$ manifolds obtained from the present $1/\kappa$ expansion, variational calculations of Rösner *et al.*,[20] and energy bounds of Handy *et al.*[21] Also given are estimates of the maximum perturbation order for which double precision arithmetic suppresses roundoff error sufficiently to yield the final coefficient with at least one significant figure.

## Figures

Fig. 1. $\boldsymbol{\mathcal{A}}_{jln}$ matrices arising from the first term in Eq.(28), for the first few orders $p$. Superscripts indicate which recursion relation, if any, of Eq.(29) may be used to compute that matrix, with preference given to Eq.(29a).

Fig. 2. Algorithm for applying recursion relations, using order $p = 8$ as an example. Displayed is the state of the matrices $\boldsymbol{\mathcal{A}}_{jln}$ after each of the following steps: (a) status after order $p - 1$; (b) apply relation Eq.(29a); (c) apply relation Eq.(29a); (d) compute three new elements. Open and filled circles denote matrices from order $p - 1$ and $p$, respectively.

Fig. 3. Nonzero elements of the first few wavefunction expansion matrices $\mathbf{a}_p$ for the ground state case.

Fig. 4. Layout of an $N \times N$ matrix on $M$ processors.

$p = 1$

$$\begin{matrix} \boldsymbol{\mathcal{A}}_{310} \\ \boldsymbol{\mathcal{A}}_{300} \end{matrix}$$

$p = 2$

$$\begin{matrix} & \boldsymbol{\mathcal{A}}_{420} \\ \boldsymbol{\mathcal{A}}_{311} & {}^{a}\boldsymbol{\mathcal{A}}_{410} \\ \boldsymbol{\mathcal{A}}_{301} & {}^{a}\boldsymbol{\mathcal{A}}_{400} \end{matrix}$$

$p = 3$

$$\begin{matrix} & \boldsymbol{\mathcal{A}}_{421} & {}^{a}\boldsymbol{\mathcal{A}}_{520} \\ \boldsymbol{\mathcal{A}}_{312} & {}^{a}\boldsymbol{\mathcal{A}}_{411} & {}^{a}\boldsymbol{\mathcal{A}}_{510} \\ \boldsymbol{\mathcal{A}}_{302} & {}^{a}\boldsymbol{\mathcal{A}}_{401} & {}^{a}\boldsymbol{\mathcal{A}}_{500} \end{matrix}$$

$p = 4$

$$\begin{matrix} & & & {}^{b}\boldsymbol{\mathcal{A}}_{630} \\ & \boldsymbol{\mathcal{A}}_{422} & {}^{a}\boldsymbol{\mathcal{A}}_{521} & {}^{a}\boldsymbol{\mathcal{A}}_{620} \\ \boldsymbol{\mathcal{A}}_{313} & {}^{a}\boldsymbol{\mathcal{A}}_{412} & {}^{a}\boldsymbol{\mathcal{A}}_{511} & {}^{a}\boldsymbol{\mathcal{A}}_{610} \\ \boldsymbol{\mathcal{A}}_{303} & {}^{a}\boldsymbol{\mathcal{A}}_{402} & {}^{a}\boldsymbol{\mathcal{A}}_{501} & {}^{a}\boldsymbol{\mathcal{A}}_{600} \end{matrix}$$

$p = 5$

$$\begin{matrix} & & {}^{b}\boldsymbol{\mathcal{A}}_{631} & {}^{a}\boldsymbol{\mathcal{A}}_{730} \\ & \boldsymbol{\mathcal{A}}_{423} & {}^{a}\boldsymbol{\mathcal{A}}_{621} & {}^{a}\boldsymbol{\mathcal{A}}_{720} \\ \boldsymbol{\mathcal{A}}_{314} & {}^{a}\boldsymbol{\mathcal{A}}_{413} & {}^{a}\boldsymbol{\mathcal{A}}_{512} & {}^{a}\boldsymbol{\mathcal{A}}_{611} & {}^{a}\boldsymbol{\mathcal{A}}_{710} \\ \boldsymbol{\mathcal{A}}_{304} & {}^{a}\boldsymbol{\mathcal{A}}_{403} & {}^{a}\boldsymbol{\mathcal{A}}_{502} & {}^{a}\boldsymbol{\mathcal{A}}_{601} & {}^{a}\boldsymbol{\mathcal{A}}_{700} \end{matrix}$$

$p = 6$

$$\begin{matrix} & & & & & {}^{b}\boldsymbol{\mathcal{A}}_{840} \\ & & & {}^{b}\boldsymbol{\mathcal{A}}_{632} & {}^{a}\boldsymbol{\mathcal{A}}_{731} & {}^{a}\boldsymbol{\mathcal{A}}_{830} \\ & \boldsymbol{\mathcal{A}}_{424} & {}^{a}\boldsymbol{\mathcal{A}}_{523} & {}^{a}\boldsymbol{\mathcal{A}}_{622} & {}^{a}\boldsymbol{\mathcal{A}}_{721} & {}^{a}\boldsymbol{\mathcal{A}}_{820} \\ \boldsymbol{\mathcal{A}}_{315} & {}^{a}\boldsymbol{\mathcal{A}}_{414} & {}^{a}\boldsymbol{\mathcal{A}}_{513} & {}^{a}\boldsymbol{\mathcal{A}}_{612} & {}^{a}\boldsymbol{\mathcal{A}}_{711} & {}^{a}\boldsymbol{\mathcal{A}}_{810} \\ \boldsymbol{\mathcal{A}}_{305} & {}^{a}\boldsymbol{\mathcal{A}}_{404} & {}^{a}\boldsymbol{\mathcal{A}}_{503} & {}^{a}\boldsymbol{\mathcal{A}}_{602} & {}^{a}\boldsymbol{\mathcal{A}}_{701} & {}^{a}\boldsymbol{\mathcal{A}}_{800} \end{matrix}$$

FIG. 1.

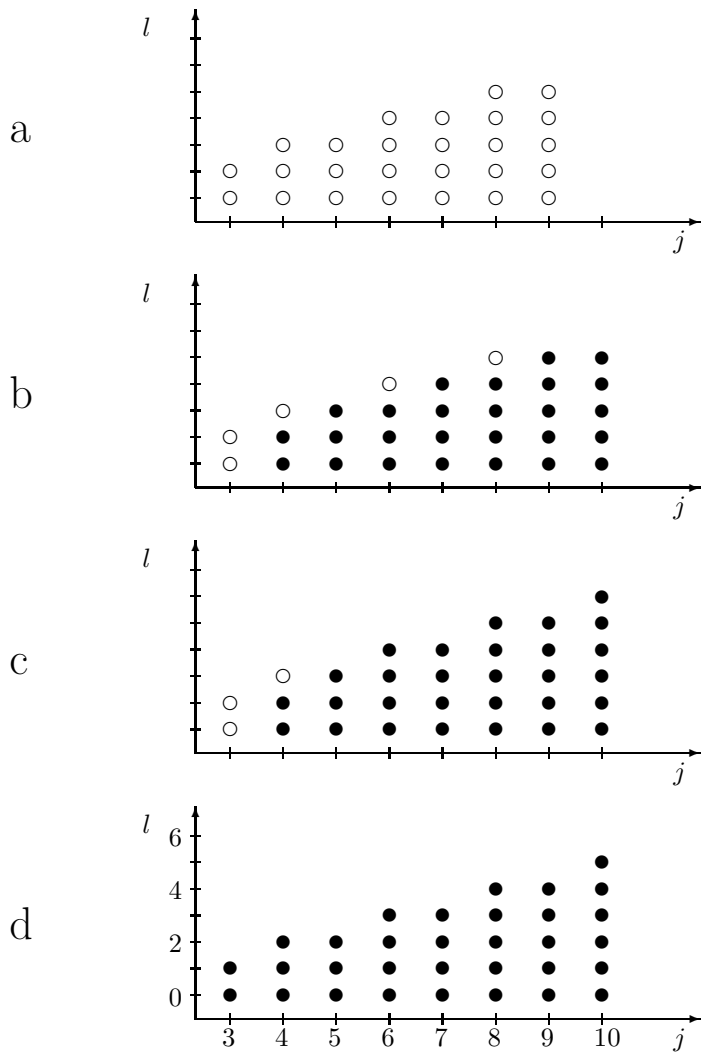


FIG. 2.

$$\mathbf{a}_0 \qquad \begin{pmatrix} \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ & & & \vdots & & & & & \ddots \end{pmatrix}$$

$$\mathbf{a}_1 \qquad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ & & & \vdots & & & & & \ddots \end{pmatrix}$$

$$\mathbf{a}_2 \qquad \begin{pmatrix} 0 & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ & & & \vdots & & & & & \ddots \end{pmatrix}$$

$$\mathbf{a}_3 \qquad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ & & & \vdots & & & & & \ddots \end{pmatrix}$$

FIG. 3.

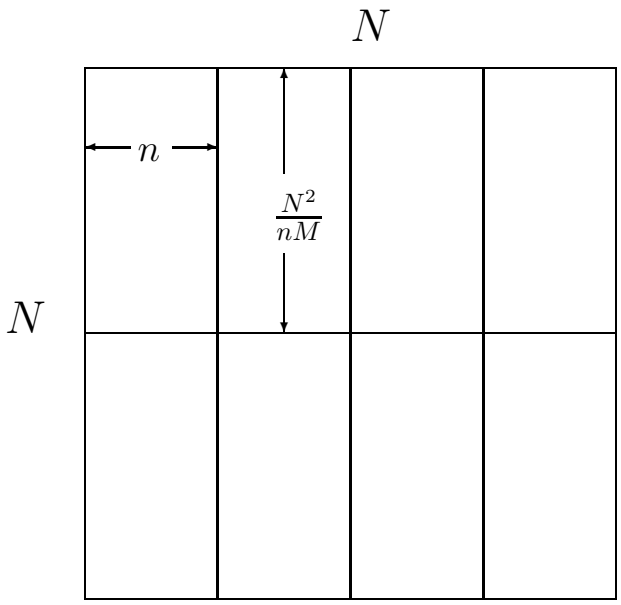


FIG. 4.